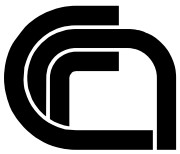


Consiglio Nazionale delle Ricerche

A Comparison of Several Algorithms for the Single Individual SNP Haplotyping Reconstruction Problem

F. Geraci

IIT TR-05/2010

Technical report

Febbraio 2010

Istituto di Informatica e Telematica

A Comparison of Several Algorithms for the Single Individual SNP Haplotyping Reconstruction Problem

Filippo Geraci

Istituto di Informatica e Telematica del CNR, Via G. Moruzzi 1, 56100-Pisa (Italy). filippo.geraci@iit.cnr.it

Abstract. Motivation: Single Nucleotide Polimorfisms are the most common form of variation in human DNA. They are involved in many research fields from molecular biology to medical therapy. The technological opportunity to deal with long DNA sequences using shotgun sequencing raised the problem of fragment recombination. In this direction the Single Individual Haplotyping (SIH) problem has received more attention in last few years.
Results: In this paper we survey seven of the most recent approaches to the SIH problem and make an extensive evaluation of them using real human haplotype data from the HapMap project. We also implemented a data generator tailored on current shotgun sequencing technology that use haplotypes from the HapMap project.
Availability: The data we used to compare the algorithms are available on demand since we think they represent an important benckmark that can be used to easily compare novel algorithmic ideas with the state of the art. Moreover, we had to reimplement six of the surveyed algorithms because the original code was not available to us. Five of these algorithms and the data generator used in this paper, endowed with a Web interface are available at http://bioalgo.iit.cnr.it/rehap.

1 Introduction

Recently the attention of researchers has shifted from what is common among the individuals of a certain species to what differ among them, thus DNA mutations. The single nucleotide polimorfism (*SNP* pronounced "snip") is the most spread form of variation in the human DNA. It consists in the variation (in a bounded range of possibilities) of the base present in a single fixed position of the DNA strand. The sequence of all the SNPs in a certain chromosome is called *haplotype*. Humans are diploid organisms. This means that, except for the sexual chromosomes of males, the chromosomes came in two copies: one inherited from the mother and one from the father. As a consequence the haplotype of a chromosome can be fully described by two sequences of SNPs: the mother's haplotype and the father's one. Since haplotypes contain all the information about DNA variation, they play a crucial role in many studies about variations in genes expression and prediction of diseases. In [**?**] the authors estimate that the overall number of SNPs in the human DNA is about 9-10 millions. A lot of effort was done by many human DNA sequencing projects that have attempted to build maps of the SNPs present in the human DNA [**?**,**?**]. At the moment the most complete map contains over 3.1 millions of SNPs.

The Single Individual Haplotype reconstruction problem (SIH) is one of the core problems for the reconstruction of whole genomes [**?**]. It consists in rebuilding the two haplotypes from a set of fragments obtained by the shotgun sequencing of the chromosomes. Actual shotgun technology produce a very large set of fragments of length in the range of 200/900 bases with a certain degree of overlap among them [**?**]. This technology does not allow to keep track of the association of a fragment with its haplotype. An important characteristic of this problem is that, unlike the fragment assembly problem, the position and orientation of the fragments is known a priori. This means that fragments can be arranged in a matrix called *SNP Matrix*. In absence of error it is easy to find a bipartition of the SNP matrix such that the fragments belonging to each partition do not conflict among them (two fragments are said to be conflicting if for a certain position not gap they have different values). In the real world application this is not the case and errors affect the SNP matrix. Errors can come from different sources. Readings errors are typically due to chemical/optical error in reading the SNP and have

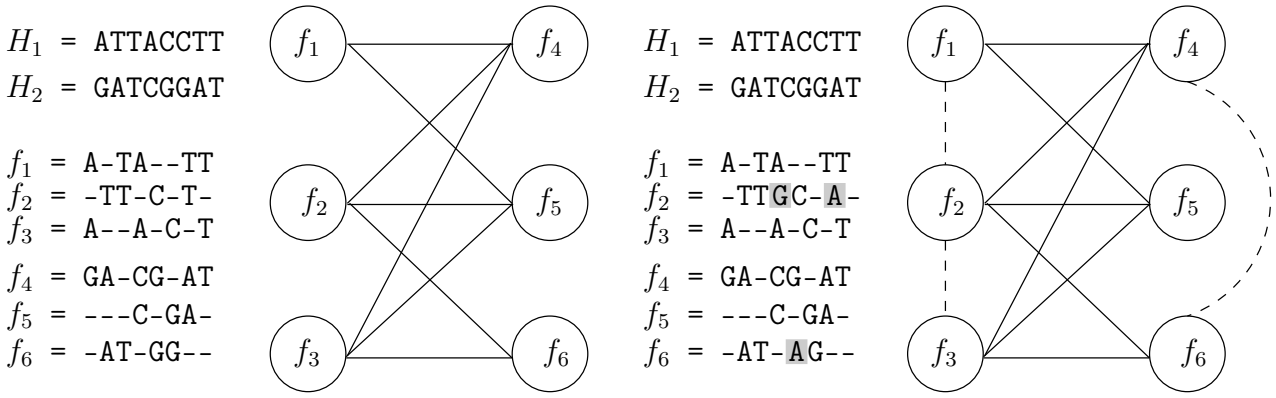
as effect the insertion of the wrong base in a certain position. Ambiguous readings happen when the signal strength of a SNP in not enough to establish with a high degree of confidence the value of a certain SNP. The effect of ambiguous readings is typically the insertion of a gap in the sequence.

In order to solve the SIH problem a number of models were introduced in the literature. Many of these models have NP-hard solutions, thus heuristics are often used. Nowadays a lot of good algorithms for the SIH problem were published, but there is not a common framework to compare all those algorithms among them. In this paper we select seven of the most effective heuristic algorithms for the SIH problem and define a common framework in which all the algorithm are evaluated. Many papers describe algorithms for the Single Individual Haplotyping problem. The corresponding code is often not available, and the papers do not describe the algorithms with enough details to carefully reimplement them. In these cases we were unable to include them in this paper. We excluded also some algorithms of which we received the software because of their characteristics. For example the Branch and Bound algorithm described in [**?**] is much slower than the competitors and its running time makes it not suitable in practice even for small datasets. The same problem is present in *HASH* [**?**]. Other algorithms like *KMec* [**?**] do not allow the insertion of mate pairs in the SNP matrix and thus conflict with the actual shotgun sequencing technology .

Our test data are available on demand, this means that they can be used in the near future to compare new ideas with the actual state of the art. Despite the high number of real human haplotypes freely available on the Web, there are no real SNP matrices. In this paper we made the effort of generating realistic data writing a software that get in input a real haplotype and simulate the actual technologies of shotgun sequencing to produce realistic SNP matrices. The SNP matrices simulator is described in section **??**.

The paper is organized as follow: in section **??** we formally define the Single Individual Haplotype Reconstruction problem and its most common computational models. In section **??** we describe all the compared algorithms. Section **??** reports quality and running time evaluation. We conclude in section **??**.

2 Problem Formulation



(a) SNP matrix without errors

(b) SNP matrix with errors

Fig. 1. On the left: two haplotype strings, six fragments (without errors) and the corresponding bipartite conflict graph. On the right: the same two haplotypes, six fragments (with errors in gray) and the corresponding conflict graph (not bipartite)

Due the diploid nature of humans' cells, except for sexual chromosomes of males, each chromosome comes in two almost identical copies, one inherited from the mother and one from the father. Actual technology of

shotgun sequencing is unable to keep track of the association between a fragment and its own chromosome. Thus, from the biological point of view the single Individual Haplotyping problem consists in the reassignment of each fragment to the original haplotype.

From the computational point of view the problem was firstly formalized in [**?**]. A fragment is represented as a string of length m such that to each character corresponds a base in the alphabet $\Sigma = \{a, c, g, t\}$ or a $-$ in case of gap. The natural way to store all n fragments is a matrix M with n rows and m columns, such that, to each row corresponds a fragment $f_i = M[i]$. The matrix M is refeared as SNP matrix. The element $M[i][j]$ stored in the j-th entry of the i-th row of M represents the j-th SNP of the haplotype for fragment f_i. We will denote this element also as $f_i[j]$. In case f_i does not cover the j-th position of the haplotype, we have a $M[i][j] = -$.

We say that a fragment f contains a gap (or is gapped) if for $i, j, k \in [1, m]$ such that $i < j < k$ we have $f[i] \neq -$, $f[j] = -$ and $f[k] \neq -$. If no fragments in M are gapped than the SNP matrix is said gapless otherwise it is said gapped.

In absence of errors in the SNP matrix, each column of M can contain one or two distinct elements. The presence of only an element indicates that the SNP in the corresponding site is homozygous otherwise the SNP is heterozygous.

We say that two fragments f_i and f_j are in conflict if exists at least a SNP position k such that both fragments have not a gap in position k and $f_i[k] \neq f_j[k]$. According with the definition of conflict between pairs of fragment, in [**?**] the conflict graph is defined. Let $G = \{V, E\}$ be a graph such that to each vertex corresponds a fragment and there is an edge between two fragments if there is a conflict between them. If M does not contains errors, then G is bipartite (See figure **??** on the left). The bipartition is not necessary unique, if the graph has several connected components. In practice, due to errors in the SNP matrix, the conflict graph is never bipartite (See figure **??** on the right). Thus, in this case, the Single Individual Haplotype reconstruction problem can be formalized as the problem of removing a certain number of edges from G until the resulting graph becomes bipartite. The problem of reducing a graph to a bipartite graph is well studied in the literature where many models were proposed. Among then in the context of the SIH problem the following formalizations are often used:

MSR **Minimum SNP Removal:** determine a minimal number of SNPs whose removal from the input set induces a bipartite graph.

MFR **Minimum Fragment Removal:** determine a minimal number of fragments whose removal from the input set induces a bipartite graph.

LHR **Longest Haplotype Reconstruction:** determine set of fragments whose removal from the input set induces a bipartite graph and the length of the induced haplotype is maximized.

MEC **Minimum Error Correction:** determine a minimal set of entries of the matrix M whose correction to a different value induces a bipartite graph.

In presence of gaps all the above problem formalization are NP-hard. More detail on the complexity of some of these problems can be found in [**?**]. It is important to observe that there is no proven relationship among the above problem formulations and the real SIH problem. Thus a more accurate solution of them not necessarily means a more precise solution of the SIH problem. That is the reason because in this paper we compared all algorithms against the ground truth instead of measuring the accuracy against the computational model.

We now introduce some definitions and notations that we will use later in the algorithms description. Let C be a set of rows of M and let $x_C[i] \in \Sigma$ the character that appear more frequently at position i among the fragments in C (or $x_C[i] = -$ if all the fragments of C have a gap in position i), we define the haplotype consensus $H(C)$ deduced by C as the string of m characters such that the character at position i is $x_C[i]$.

We define the generalized Hamming distance between two fragments f_i and f_j as:

$$Ham(f_i, f_j) = \sum_{k=1}^{m} ham(f_i[k], f_j[k]) \tag{1}$$

where

$$ham(f_i[k], f_j[k]) = \begin{cases} 1 \text{ if } f_i[k] \neq f_j[k] \neq - \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

If the generalized Hamming distance between two fragments is different from 0, we say that the two fragments are in conflict.

3 Algorithms

3.1 Clustering algorithm for the MEC model

In [**?**] a clustering algorithm is used to split the rows of M in two sets. The main contribution of the paper is given by the combination of the two distance functions used by the clustering algorithm. As first distance the authors use the generalized Hamming distance Ham as defined in equation (**??**). This distance takes into account only the number of mismatch between two fragments. The second distance is defined as follows: let f_i and f_j be two fragments:

$$D'(f_i, f_j) = \sum_{k=1}^{m} d'(f_i[k], f_j[k]) \quad (3)$$

where

$$d'(f_i[k], f_j[k]) = \begin{cases} -1 \text{ if } f_i[k] = f_j[k] \neq - \\ 1 \text{ if } f_i[k] \neq f_j[k] \neq - \\ 0 \text{ otherwise} \end{cases}$$

The definition of distance in equation (**??**) also takes into account the number of match between the two fragments. This means that, given a certain fixed number of mismatch between two fragments, the more they overlaps the more they are close according with D'. Note that D' is not a distance in a strict sense, in fact it can be negative and has values in the range $[-m, m]$. Moreover the triangular inequality does not hold.

Using the above distance functions, the authors propose a simple iterative clustering procedure. In order to compare two distances, the functions Ham and D' are evaluated in cascade.

The algorithm proceeds as follows: (1) for each possible pair of fragments in the SNP matrix the generalized Hamming distance is computed. let f_i and f_j be the two furthest fragments according with Ham, we initialize the two sets $C_1 = f_i$ and $C_2 = f_j$. (2) Let $H_1 = H(C_1)$ and $H_2 = H(C_2)$ be the two consensus strings derived from C_1 and C_2, all the fragments are compared with H_1 and H_2 and assigned to the corresponding closer set. In case a fragment is equidistant from the two consensus strings, the distance D' is used to decide to which set assign the fragment. (3) Once all fragments are assigned, the consensus strings H_1 and H_2 are updated and the algorithm restart from (2). The procedure loops until a stable haplotype pair is found.

3.2 Clustering algorithm for the MLF problem

In [**?**] the authors raise a weighted variant of the MEC problem called WMLF (Weighted Minimum Letter Flip). Suppose to have a matrix W such that each entry is a number in the range $[0, 1]$ representing the degree of confidence of the SNP in the same position in the matrix M. We can define a waighted version of the generalized Hamming distance between two fragments f_i and f_j as:

$$WHam(f_i, f_j) = \sum_{k=1}^{m} wh(f_i[k], f_j[k])$$

where

$$wh(f_i[k], f_j[k]) = \begin{cases} \min(W[i][k], W[j][k]) \text{ if } f_i[k] \neq f_j[k] \neq - \\ 0 \text{ otherwise} \end{cases}$$

The distance $WHam$ is extended to deal with haplotypes, assuming that the weight associated to each character of the consensus string is 1.

The proposed algorithm is based on the well-known one-pass k-means clustering algorithm due to McQueem [**?**]. The procedure initialization consists in randomly partitioning the rows of M in two sets C_1 and C_2 deriving from them two consensus strings: $H_1 = H(C_1)$ and $H_2 = H(C_2)$. In the main procedure loop, at each iteration C_1 and C_2 are reinitializated and a new partition of the rows in M is done. For each fragment f_i, we compare its distance from H_1 and H_2. If $WHam(f_i, H_1) > WHam(f_i, H_1)$ then f_i is assigned to C_1, otherwise the fragment is assigned to C_2. The procedure terminates when two stable haplotypes are detected. More details about the convergence of this method can be found in [**?**]. Due to the random initialization of the sets C_1 and C_2, every run of this algorithm on a certain dataset can return different haplotype consensus. In order to mitigate the effect of randomness the authors run the algorithm 100 times and return as final result the consensus pair that minimize the following target function:

$$F(C_1, C_2) = \sum_{k=1}^{2} \sum_{f \in C_k} WHam(f, H(C_k)) \tag{4}$$

The main drawback of this algorithm stands is that, with actual shotgun sequencing technology, the informations about the confidence level of each fragment are typically not available. In this case the algorithm have to assume the same level of confidence for each fragment, thus $WHam$ reduces to Ham and the target function to minimize in equation (**??**) reduces to the target function of the MEC model. Even in [**?**] experiments are made assuming each entry of W as equal to 1.

3.3 Fast Hare

Fast Hare [**?**] is one of the first heuristic for the SIH problem. Despite it was designed to work in a gapless environment, our experiments and those reported in [**?**] confirm that this method still works nicely also in the more general case in which gaps are allowed. As preliminary step, Fast Hare removes from the SNP matrix M all those columns such that there is a character (not $-$) that is over-represented with respect to the others. These colums are considered as homozigous sites. We call this reduced matrix $\hat{M}$. More formally: let $|x_c|$ the number of occurrences of the character $x \in \Sigma$ in the column c of M. The probability to find x in c is:

$$P_c(x) = \frac{|x_c|}{\sum_{\sigma \in \Sigma} |\sigma_c|}$$

If it exists a character x in c such that $P_c(x) >= t$, the column c is removed from the SNP matrix and the character x will be inserted in the final solution. In [**?**] and in our experiments the threshold t is set to 0.8. The intuition behind this filtering comes from the fact that when $P_c(x) >= t$ column c represents a homozigous SNP (thus the column does not help in the reconstruction procedure) and the positions (not gaps) not containing x are errors. As the first step Fast Hare sorts the fragments of $\hat{M}$ according to the following ordering criterion: let k_i and k_j be respectively the position of the first character not gap in f_i and f_j, thus $k_i \leq k_j \Rightarrow f_i \preceq f_j$. After sorting the fragments of $\hat{M}$, Fast Hare initialize two sets $C_1 = C_2 = \emptyset$. At this point, according to their order, the fragments of $\hat{M}$ are scanned one at time. The first fragment is assigned to C_1. Considering the fragment f_i, Fast Hare computes its similarity with the two partial consensus strings $H(C_1)$ and $H(C_2)$. As similarity score Fast Hare uses $-D'$ where D' is the distance we defined in equation (**??**)). The fragment is assigned to the set whose consensus shows higher similarity. Note that the values of $-D'$ stand in the range $[-1, 1]$ and for all fragments holds $-D'(f, H(\emptyset)) = 0$.

3.4 SpeedHap

SpeedHap [?] [?] approaches the haplotype assembly problem differently from previously described algorithms. Instead of considering entirely the fragments, it attempts to solve n instances of the haplotyping problem on 1-base long fragments and combines results.

SpeedHap is a greedy heuristic. It builds its solution in a pre-processing phase and three main phases. The goal of each phase is to exploit the outcome solution of the preceeding phase and improve it by relaxing some constraints.

In the pre-processing phase SpeedHap performs a statistical analysis of the columns of the SNP matrix attempting to locate detectable errors and, if possible, correct them. In this phase the heuristic also set up some data structures.

In the first phase the heuristic selects an initial set of columns such that they are likely to contain as few errors as possible. For each column SpeedHap builds a set G_i (called *profile*) in which each element is the set of all the indices of the fragments containing the same character in position i. It is easy to observe that only profiles having two elements (i.e columns of the SNP matrix containing exactly two distinct characters) are of interest since an empty profile corresponds to a hole in the haplotype, a profile with just an element corresponds to a homozygous site, a profile with more than two characters must contains errors. Let $P_i = (P_i(1), P_i(2))$ be the profile of column i such that it corresponds to a heterozygous site. Given two columns i and j, we can define the *error matrix* as:

$$E_{i,j} = \begin{pmatrix} P_i(1) \cap P_j(1) & P_i(1) \cap P_j(2) \\ P_i(2) \cap P_j(1) & P_i(2) \cap P_j(2) \end{pmatrix}$$

When the error matrix has positive values only in one diagonal and it is of full rank, there are no detectable conflicts between the two involved columns. Now, consider a graph such that it has a vertex for each column of M corresponding to a heterozygous site and there is an edge between two vertices if the corresponding error matrix does not reveals inconsistencies (i.e. the matrix is diagonal and of full rank) Using a DFS search we can partition the graph in connected components. To each component corresponds a set of columns not conflicting among them. The initial partitioning of the fragments of M is extracted from the largest set. In the second phase the algorithm works in a similar manner. The partitioning obtained from the previous step acts as a special profile (pivot). All the columns not involved in the previous phase are compared with the pivot and the error matrix is computed. If the error matrix does not shows inconsistencies, the column is included in the solution. This procedure is repeated iteratively until it is no longer possible to add new columns to the pivot. In the last phase some constraints are relaxed and to be admitted to be part of the solution does not longer requires an error matrix of full rank.

Another important contribution of SpeedHap is the use of the context for resolving ambiguities in the final haplotypes reconstruction. In [?] the authors parse a large database of human haplotypes measuring the empirical entropy of order up to 2. As result they show that there seems to exist a statistical correlation among the base in a certain SNP site, its preceeding SNP site and its succeeding one.

When the coverage is low and the error rate is relatively high, it is not infrequent the case in which, building a haplotype, exactly half of the fragments in a certain position have a certain value and half of them have another value. In this case the choice of which character should appear in the haplotype is arbitrary. To break ties, SpeedHap exploits the statistical correlation among contiguous sites. Consider the case in which the procedure have to choice for the site in position i whether select A or B. Let x be the character in position $i-1$ and z the character in position $i+1$. The procedure will decide for: A if the empirical entropy of the string xAz is lower than those of the string xBz, otherwise it will decide for B.

3.5 HapCUT

HapCUT [?] approaches the haplotype assembly as a MAX-CUT problem. The HapCUT algorithm considers the submatrix of the SNP matrix in which we remove all the columns corresponding to Homozygous SNPs and

all the columns in which there are present more than two distinct bases (i. e. there must be at least a mistaken base). Let us call the resulting matrix X. Due to the fact that each column of X contains exactly two possible SNP values, it can be represented using the restricted alphabet $0, 1, -$. The haplotype pair H associated to X is composed by a binary string h and its bit-wise complement $\hat{h}$.

Given a certain haplotype pair H the authors define a graph $G_X(H)$ such that there is a vertex for each column of the matrix X and there is an edge between two vertices of $G_X(H)$ if the corresponding columns in X are linked by at least one fragment. Consider the fragment X_i such that it covers both positions j and k. Let $X_i[j,k]$ and $H[j,k]$ represent the restriction of X_i and H to loci i and j. There are two cases: $X_i[j,k]$ match one of the two haplotype strings of $H[j,k]$, or $X_i[j,k]$ does not match any. The weight $w_H(j,k)$ associated to the edge between node j and k in the graph $G_X(H)$ is given by the number of fragments such that $X_i[j,k]$ does not match any string in $H[j,k]$ minus the number of fragment such that the match exists. The higher is $w_H(j,k)$, the weaker is correlation between the haplotype pair H and the SNP matrix restricted to columns j and k. Let $(S, X-S)$ be a cut of G, the weight of the cut is defined as follows:

$$w_H(S) = \sum_{j \in S, k \in X-S} w_H(j,k)$$

Consider the haplotype pair H_S derived from H by flipping all the elements involved in S. The authors show that if $w_H(S)$ is positive for the graph $G_X(H)$ then the following holds:

$$\mathrm{MEC}(H_S) = \mathrm{MEC}(H) - w_H(S) > \mathrm{MEC}(H)$$

In consequence of the above result, the problem of finding a haplotype pair minimizing the MEC score reduce to the problem of finding a max-cut in $G_X(H)$. This problem is well known to be NP-complete [**?**], thus heuristic methods are often used.

The HapCUT procedure exploits the connection between the MAC optimization and the max-cut problem. Starting from a random haplotype pair, HapCUT iteratively attempts to refine the haplotype pair to reduce the MEC score. At each iteration, the algorithm proceeds as follows: (1) compute the graph $G_X(H)$, (2) compute a max-cut S using a greedy heuristic like that in [**?**], (3) if the MEC score of the pair H_S is smaller than the score of H, keeps as new haplotype pair H_S.

The procedure loops until is no longer possible to reduce the MEC score.

3.6 DGS

In [**?**] the authors describe a huge work about genome sequencing. In the paper they also describe a good algorithm for the SIH problem. For lack of a better name we call this algorithm *DGS*. Like in HapCUT, this algorithm works with a submatrix of M in which we remove all the columns corresponding to Homozygous sites and those with more than two distinct values. Let us call again this matrix X. Even in this case X can be represented using the restricted alphabet $0, 1, -$. The haplotype pair H associated to X is composed by a binary string h and its bit-wise complement $\hat{h}$. The DGS procedure works in two phases: an initialization in which we build a pair of initial haplotypes and a refinement step in which haplotypes are iteratively refined. The initialization works as follow:

1 the fragment with the minimal number of gaps is used to initialize an haplotype. The other haplotype is initialized with the complementary string;
2 until no more fragment share non-missing information with an haplotype, select the fragment such that the number of columns it has in common with one haplotype minus number of columns indicating the other haplotype is maximal and assign it to the corresponding haplotype. The other haplotype is updated with the complementary string;

The second phase iteratively refine the haplotype consensus strings and stops when, at the end of an iteration, the solution no longer change. At each iteration: the haplotype consensus strings are determined by majority rule, then each fragment is associated to the closest haplotype.

	Algo	e=0.0				e=0.1				e=0.2				e=0.3			
		c=3	c=5	c=8	c=10	c=3	c=5	c=8	c=10	c=3	c=5	c=8	c=10	c=3	c=5	hc=8	c=10
l=100	Baseline	1.000	1.000	1.000	1.000	0.971	0.992	0.997	0.999	0.898	0.944	0.967	0.980	0.787	0.840	0.878	0.903
	SpeedHap	**0.999**	**1.000**	**1.000**	**1.000**	0.895	0.967	**0.989**	0.990	0.623	0.799	0.852	0.865	0.480	0.637	0.667	0.676
	Fast Hare	**0.999**	**0.999**	**1.000**	**1.000**	0.919	0.965	**0.993**	**0.998**	0.715	0.797	0.881	0.915	0.617	0.639	0.661	0.675
	2d-mec	0.990	**0.997**	**1.000**	**1.000**	0.912	0.951	0.983	0.988	0.738	0.793	0.873	0.894	**0.623**	0.640	0.675	0.678
	HapCUT	**1.000**	**1.000**	**1.000**	**1.000**	**0.929**	0.920	0.901	0.892	**0.782**	**0.838**	0.864	0.871	0.602	0.629	0.673	0.709
	MLF	0.973	0.992	**0.997**	**0.998**	0.889	0.970	0.985	**0.995**	0.725	**0.836**	**0.918**	**0.938**	0.618	**0.653**	**0.697**	**0.715**
	SHR	0.816	0.861	0.912	0.944	0.696	0.738	0.758	0.762	0.615	0.655	0.681	0.699	0.557	0.599	0.632	0.632
	DGS	**1.000**	**1.000**	**1.000**	**1.000**	**0.930**	**0.985**	**0.989**	**0.997**	0.725	0.813	0.878	0.917	0.611	0.647	0.663	0.688
l=350	Baseline	1.000	1.000	1.000	1.000	0.978	0.990	0.997	0.999	0.896	0.943	0.968	0.981	0.783	0.840	0.873	0.903
	SpeedHap	**0.999**	**1.000**	**1.000**	**1.000**	0.819	0.959	0.984	0.984	0.439	0.729	0.825	0.855	0.251	0.578	**0.629**	0.638
	Fast Hare	0.990	**0.999**	**1.000**	**0.999**	0.871	0.945	0.985	**0.995**	0.684	0.746	0.853	**0.877**	**0.590**	0.602	0.626	0.644
	2d-mec	0.965	0.993	**0.998**	**0.999**	0.837	0.913	0.964	0.978	0.675	0.729	0.791	0.817	**0.593**	**0.606**	0.623	0.634
	HapCUT	**1.000**	**1.000**	**1.000**	**1.000**	**0.930**	0.913	0.896	0.888	**0.771**	**0.831**	**0.862**	0.867	0.565	0.582	0.621	**0.664**
	MLF	0.864	0.929	0.969	0.981	0.752	0.858	0.933	0.962	0.642	0.728	0.798	0.831	0.581	**0.606**	**0.634**	0.641
	SHR	0.830	0.829	0.895	0.878	0.682	0.724	0.742	0.728	0.591	0.632	0.670	0.668	0.548	0.557	0.604	0.619
	DGS	**0.999**	**1.000**	**1.000**	**1.000**	**0.926**	**0.978**	**0.996**	**0.998**	0.691	0.769	0.842	**0.878**	0.578	**0.609**	0.628	0.641
l=700	Baseline	1.000	1.000	1.000	1.000	0.971	0.991	0.997	0.999	0.898	0.942	0.966	0.980	0.786	0.838	0.875	0.902
	SpeedHap	**0.999**	**1.000**	**1.000**	**1.000**	0.705	0.947	**0.985**	0.986	0.199	0.681	0.801	0.813	0.095	0.523	**0.616**	**0.627**
	Fast Hare	**0.988**	**0.999**	**1.000**	**0.999**	**0.829**	0.949	**0.986**	**0.995**	0.652	0.712	0.808	**0.872**	**0.581**	0.591	0.615	0.616
	2d-mec	0.946	0.976	0.992	**0.997**	0.786	0.880	0.948	0.965	0.647	0.697	0.751	0.778	**0.583**	**0.596**	**0.613**	0.622
	HapCUT	**1.000**	**1.000**	**1.000**	**1.000**	**0.927**	0.916	0.896	0.889	**0.753**	0.825	**0.856**	0.861	0.552	0.555	0.597	0.645
	MLF	0.787	0.854	0.919	0.933	0.698	0.809	0.863	0.884	0.624	0.682	0.747	0.765	0.570	**0.594**	**0.614**	**0.625**
	SHR	0.781	0.832	0.868	0.898	0.668	0.716	0.743	0.726	0.591	0.617	0.653	0.675	0.536	0.562	0.611	**0.625**
	DGS	**0.999**	**1.000**	**1.000**	**1.000**	**0.931**	**0.977**	**0.987**	**0.997**	0.669	**0.741**	0.818	0.861	0.573	0.595	**0.614**	0.622

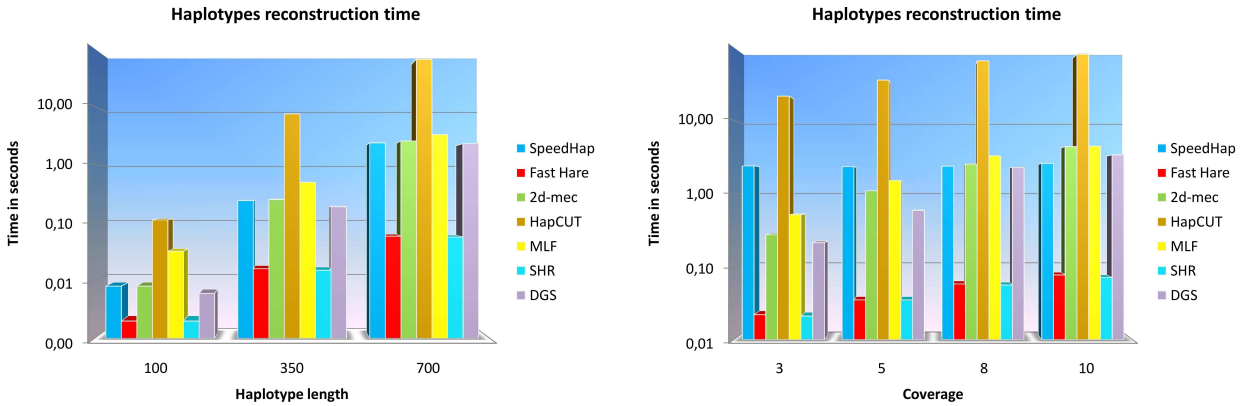
Table 1. Each entry of the table represents the average, over 100 randomly selected HapMap strings, of the Error Rate when the Hamming distance is in the range $[0.7m, m]$. The free parameters are: 1) the haplotype length $l = 100; 350; 700$; 2) the coverage $c = 3; 5; 8; 10$; 3) and the errors rate $e = 0\%; 10\%; 20\%; 30\%$. In **bold** the algorithms with highest performance, in gray the algorithms with the second best performance. We consider as equal the performance of two algorithms when the difference between their error rate is in the range of 0 to 0.005.

3.7 SHR-three

In [**?**] and [**?**], the authors propose a probabilistic framework to approach the SIH problem. According to the proposed model, the authors design a novel probabilistic algorithm and generalize it to handle reading errors and gaps. The most general variant of this algorithm is called *SHR-three*. The algorithm requires as input the SNP matrix and a parameter u that controls the number of iterations made by the main loop. As made by the authors in their experiments, we set $u = 10$.

The *SHR-three* main loop is as follows: (1) select at random two fragments f_1 and f_2 from M and assign to each of them an empty set (C_1 to f_1 and C_2 to f_2) (2) each fragment of M is compared through the generalized Hamming distance with f_1 and f_2 and inserted in the set related to the closest fragment (3) for each of the two sets compute the MEC score (i.e. the sum of the distances among each fragment in C_i and f_i for $i = \{1, 2\}$) and get as score the highest value. (4) if the computed score is lower than the previous computed ones, than $\hat{C_1} = C_1$ and $\hat{C_2} = C_2$.

As consensus strings *SHR-three* returns $H(\hat{C_1})$ and $H(\hat{C_2})$.



(a) Variable haplotype length, $c = 8$

(b) Variable coverage, $l = 700$

Fig. 2. Average running time expressed in seconds over 100 instances for different settings of the halpotype length (left) and coverage (right). The error rate is set to 0.2.

4 Discussion

4.1 Input data and fragment generation

The research project *HapMap* [**?**] has produced a map of the human haplotypes that is now publicly available [**?**,**?**]. The available data are about the haplotypes of all 22 chromosomes of 209 different individuals (half males and half females) coming from 4 different populations. For females also the haplotypes of the X chromosome is available. Thus we were able to generate the fragments and the SNP matrices from real data instead of using as input synthetic haplotypes. Using real haplotypes the Hamming distance between them is no longer a free parameter. We observed that haplotype pairs show a great variability in the Hamming distance. Typical values of Hamming distance are in the range $[0.4m, m]$. In order to evaluate if Hamming distance produces effects in the outcome of the tested algorithms, we made two set of experiments: one in which we select haplotypes pairs having Hamming distance less than $0.7m$ and one such that the Hamming distance between the considered haplotypes is greater than $0.7m$

For the extraction of the SNP matrix from the haplotypes we used the method described in [**?**] taking in account standard parameters in current technology for shotgun sequencing. According to [**?**], current technology is able to manage DNA fragments of hundreds of bases and the average distance in bps of two SNPs in human DNA is quantified in 300 bps on average. Thus each DNA fragment cover roughly a number of SNPs in the range $[3, 7]$.

Given a pair of haplotypes of length l, the generation of the SNP matrix is as follow: each haplotype is replicated c times, then each copy is broken in non overlapping fragments whose size is in the range $[3, 7]$. According to a certain probability some fragments are merged again in order to simulate matepair sequences (In our experiments, at the end of this phase globally 50% of the fragments are 1-gapped). We then arrange the fragments in a matrix and insert errors according with a uniform distribution. Note that the number of fragments is not determined a priori but it depends on the length l, on the coverage c and on the distribution of the fragment lengths.

4.2 Quality evaluation

In order to evaluate the quality of the tested algorithm we use a slightly modified version of the well known error rate. Let $H = (h_1, h_2)$ be the pair of correct haplotypes each of length m. Let $\hat{H} = (\hat{h}_1, \hat{h}_2)$ be the pair of

consensus haplotypes returned by an algorithm. The reconstruction rate is:

$$RR_{\hat{H},H} = \frac{\min(D(h_1,\hat{h}_1) + D(h_2,\hat{h}_2), D(h_1,\hat{h}_2) + D(h_2,\hat{h}_1))}{2m}$$

In the standard reconstruction rate formulae D is the generalized Hamming distance, in our case D is defined as follows:

$$D(h_i,\hat{h}_j) = \sum_{k=1}^{m} d(h_i[k],\hat{h}_j[k])$$

where

$$d(h_i[k],\hat{h}_j[k]) = \begin{cases} 0 \text{ if } h_i[k] = \hat{h}_j[k] \neq - \\ 1 \text{ otherwise} \end{cases}$$

In practice we modified the Hammimng distance in a way that gaps receive the same penalty of errors. This choice is driven by the fact that the consensus string in which each position is gap is favoured by the standard reconstruction rate, instead we want to penalize it.

Table **??** reports the reconstruction rate of all the algorithms for the case in which the average Hamming distance between the input haplotypes is higher than $0.7m$. We do not report the results for the case in which the Hamming distance is lower since the performances (both in terms of reconstruction rate and running time) of the algorithms are similar.

In order to compare all the algorithms with the optimal haplotype reconstruction, table **??** also reports the reconstruction rate for the naive baseline algorithm that can access the true fragment bipartition and simply reconstruct haplotypes by majority.

As shown in table **??**, when the error rate is low (up to 0.1) the DGS algorithm performs permanently better then the others. For higher error rate there is not an algorithm that works clearly better then the others. For small fragments (with $l = 100$) and coverage higher than 3 MLF outperforms the others. For the other cases there is not strong winner. If we consider the best and the second best result (highlighted values in table **??**), we observe that Fast Hare should be considered reliable for error rate up to 0.2. In the case in which the haplotypes length is set to 100, MLF can be considered the most reliable algorithm. When the haplotypes length is set to 350 bps, the most reliable algorithm is DGS followed by Fast Hare. It is possible to observe that for low error rate DGS is always among the best algorithms. For low error rate SpeedHap performs quite good, while for high error rate MLF becames reliable. The case in which the haplotypes length is set to 700, is similar to the previous case. The DGS algorithm is reliable in all settings and outperforms the other algorithms for low error rate. Even in this case Fast Hare is the second best algorithm.

Looking at the MLF performances, it can be observed that the higher the error rate, the better works the strategy of computing many independent haplotype pairs and return the solution that minimize the MEC score.

It is surprising that, even in the cases in which the error rate is set to 0, the 2d-mec, MLF and SHR algorithms can introduce errors in their final solution. This can be explained by the fact that their initializations involve random choices that can heavily affect the final result. The other algorithm are almost always able to rebuild entirely the haplotypes without errors (sometime introducing some gap).

It should be observed that the SHR algorithm consistently has a reconstruction rate lower then the other algorithms. This can be imputed to the initialization step of the algorithm in which the two sets C_1 and C_2 are initilazied with two random fragments that have high probability of being mate in the correct bipartition.

Another important observation is that when the error rate is 0.3 and the coverage is 3, the SpeedHap algorithm performs much worse than in the other cases. This is due to the fact the high error rate and low coverage makes the first phase of SpeedHap to be unable to select the set of columns that are likely to contains few errors. The effect of this phenomenon is that SpeedHap is unable to assign the most of the rows hence a wide part of the haplotypes is filled with gaps.

Looking at the algorithms that compute a certain number of solutions and return the one that explicitly minimize the MEC score (MLF and SHR) we can observe that their performances are not among the best ones.

According with this observation it seems that the target function of minimizing the MEC score, performs worse than other approachs. We do not want to claim that this observation is necessarily true, in fact (even if not explicitly) DGS minimize the MEC score and attain good results.

4.3 Running time evaluation

In this section we show the running time of the compared algorithms. Except for HapCUT whose implementation is freely provided by the authors, we reimplemented all the other algorithms using Python v2.6. For each parameters assignment we run the algorithms over 100 different instances and collect the average running time. For our tests we used a Pentium D 3.2 Ghz endowed with 3 Gb of RAM.

Figure **??** shows a comparison of the haplotypes reconstruction time. The slowest algorithm is HapCUT while the two fastest algorithms are Fast Hare and SHR. Except for HapCUT, it is possible to observe that all the algorithms are able to solve all the instances of the reconstruction problem for each parameter assignment in less then 5 seconds in the worst case making all them suitable for real applications. Instead HapCUT running time does not scale and requires tens of seconds for large instances of the reconstruction problem.

Figure **??** shows that both haplotypes length and coverage affect the final running time of all the algorithms in different ways. Coverage involves a linear increase of the running time for all algorithms, while haplotypes length involves a quadratic increase. For lack of space, in this paper we do not report results in the case in which we vary the error rate or the Hamming distance because we observed that they have no effect on the running time of all the algorithms.

5 Conclusion

The Single Individual Haplotyping problem is one of the core problems in the whole genome sequencing. Due to the presence of various types of errors and missing data in the fragments the problem is very hard to solve. In recent years many algorithms and heuristics were proposed in the literature, but a systematic comparison among them is missing. In this paper we survey seven algorithms among the most commonly used for the Single Individual Haplotyping problem. We also developed a common framework to compare them. Our framework simulate the actual technology for shotgun sequencing to generate realistic SNP matrices from real human haplotypes collected from the HapMap project. The data we used for the comparison are available upon request thus can be used in the near future to compare novel algorithmic ideas with the actual state of the art.

Funding

This research has been partially supported by the Italian Registry of ".it" ccTLD and by the EU funded 7FP Virtual Physiological Human Network of Excellence (VPH NoE) (contract number 223920).

Acknowledgements

We would like to thank Rui-Sheng Wang, Jianxin Wang, Minzhu Xie, Zhixiang Chen and Zhiyu Zhao that provided us their software and allowed us to test it.

References

1. V. Bansal, A. L. Halpern, N. Axelrod, and V. Bafna. An mcmc algorithm for haplotype assembly from whole-genome sequence data. *Genome Research*, 2008.

2. Vikas Bansal and Vineet Bafna. Hapcut: an efficient and accurate algorithm for the haplotype assembly problem. In *ECCB*, pages 153–159, 2008.
3. Zhixiang Chen, Bin Fu, Robert T. Schweller, Boting Yang, Zhiyu Zhao, and Binhai Zhu. Linear time probabilistic algorithms for the singular haplotype reconstruction problem from snp fragments. *Journal of Computational Biology*, 15(5):535–546, 2008.
4. Zhixiang Chen, Bin Fu, Robert T. Schweller, Boting Yang, Zhiyu Zhao, and Binhai Zhu. Linear time probabilistic algorithms for the singular haplotype reconstruction problem from snp fragments. In *APBC*, pages 333–342, 2008.
5. Rudi Cilibrasi, Leo van Iersel, Steven Kelk, and John Tromp. On the complexity of the single individual SNP haplotyping problem. *Algorithmica*, 2007. In print.
6. Frazer. A second generation human haplotype map of over 3.1 million snps. *Nature*, 449:851–861, October 2007.
7. L.M. Genovese, F. Geraci, and M. Pellegrini. Speedhap: An accurate heuristic for the single individual snp haplotyping problem with many gaps, high reading error rate and low coverage. *EEE/ACM Transactions on Computational Biology and Bioinformatics*, 2008.
8. Loredana M. Genovese, Filippo Geraci, and Marco Pellegrini. A fast and accurate heuristic for the single individual snp haplotyping problem with many gaps, high reading error rate and low coverage. In *WABI*, pages 49–60, 2007.
9. HapMap. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.
10. R. M. Karp. Reducibility among combinatorial problems. pages 85–103, 1972.
11. Giuseppe Lancia, Vineet Bafna, Sorin Istrail, Ross Lippert, and Russell Schwartz. SNPs problems, complexity, and algorithms. In *Proceedings of the 9th Annual European Symposium on Algorithms*, pages 182–193. Springer-Verlag, 2001.
12. Samuel Levy, Granger Sutton, Pauline C Ng, Lars Feuk, Aaron L Halpern, Brian P Walenz, Nelson Axelrod, Jiaqi Huang, Ewen F Kirkness, Gennady Denisov, Yuan Lin, Jeffrey R MacDonald, Andy Wing Chun Pang, Mary Shago, Timothy B Stockwell, Alexia Tsiamouri, Vineet Bafna, Vikas Bansal, Saul A Kravitz, Dana A Busam, Karen Y Beeson, Tina C McIntosh, Karin A Remington, Josep F Abril, John Gill, Jon Borman, Yu-Hui Rogers, Marvin E Frazier, Stephen W Scherer, Robert L Strausberg, and J. Craig Venter. The diploid genome sequence of an individual human. *PLoS Biol*, 5(10):e254, 09 2007.
13. Lei Li, Jong Hyun Kim, and Michael S. Waterman. Haplotype reconstruction from SNP alignment. In *Proceedings of the seventh annual international conference on Computational molecular biology*, pages 207–216. ACM Press, 2003.
14. J. McQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Mathematics, Statistics, and Probability*, pages 281–298, 1967.
15. Olena Morozova and Marco A. Marra. Applications of next-generation sequencing technologies in functional genomics. *Genomics*, 92(5):255 – 264, 2008.
16. Gene Myers. A dataset generator for whole genome shotgun sequencing. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 202–210. AAAI Press, 1999.
17. Alessandro Panconesi and Mauro Sozio. Fast hare: A fast heuristic for single individual SNP haplotype reconstruction. In *WABI*, pages 266–277, 2004.
18. Sachidanandam. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. In *Nature 409*, pages 928–933, February 2001.
19. Sartaj Sahni and Teofilo Gonzalez. P-complete problems and approximate solutions. *Annual Symposium on Switching and Automata Theory*, 0:28–32, 1974.
20. Rui-Sheng Wang, Ling-Yun Wu, Zhen-Ping Li, and Xiang-Sun Zhang. Haplotype reconstruction from snp fragments by minimum error correction. *Bioinformatics*, 21(10):2456–2462, 2005.
21. Ying Wang, Enmin Feng, and Ruisheng Wang. A clustering algorithm based on two distance functions for mec model. *Computational Biology and Chemistry*, 31(2):148–150, 2007.
22. Minzhu Xie, Jianxin Wang, and Jianer Chen. A practical exact algorithm for the individual haplotyping problem mec. In *BMEI '08: Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics*, pages 72–76, Washington, DC, USA, 2008. IEEE Computer Society.
23. Yu-Ying Zhao, Ling-Yun Wu, Ji-Hong Zhang, Rui-Sheng Wang, and Xiang-Sun Zhang. Haplotype assembly from aligned weighted SNP fragments. *Computational Biology and Chemistry*, 29(4):281–287, 2005.
24. YuZhong Zhao, Yun Xu, Qiangfeng Zhang, and Guoliang Chen. An overview of the haplotype problems and algorithms. *Frontiers of Computer Science in China*, 1(3):272–282, 2007.